[illegible]

shifting all of said bits of said at least one short immediate value using said op-code and only said portion of bits to produce a shifted immediate value; and

storing said shifted immediate value in a register.

32. The digital processor of Claim 31, wherein said at least one instruction word having a plurality of fields and said at least one instruction word having a short immediate value comprise the same instruction word(s).

REMARKS

Claims 1-24 were pending in the application. By this paper, Applicant has added new Claims 25-32. Accordingly, Claims 1-32 are presented herein for examination.

New Claims 25-32 relate generally to subject matter incorporated by reference into the present application from Applicant's co-pending U.S. Patent Application Serial No. 09/524,178 filed March 13, 2000 and entitled "Method and Apparatus for Loose Register Encoding Within a Pipelined Processor".

Applicant has also amended the specification to (i) include appropriate copyright notice; (ii) correct the priority claim of the application, and (iii) include information regarding related applications. Applicant submits that these amendments introduce no new matter.

Replacement ("clean") pages for the foregoing amendments and additions are provided herewith.

Applicant hereby specifically reserves the right to prosecute claims of different or broader scope, including those cancelled without prejudice herein, in a continuation or divisional application.

Applicant notes that any amendments, cancellations, or additions made herein are made solely for the purposes of more clearly and particularly describing and claiming the invention, and not for purposes of overcoming art or for reasons relating to patentability. The Examiner should infer no (i) adoption of a position with respect to patentability, (ii) change in the Applicant's position with respect to any claim or subject matter of the invention, or (iii) acquiescence in any way to any position taken by the Examiner, based on such amendments, cancellations, or additions.

If the Examiner has any questions or comments which may be resolved over the telephone, he is requested to call the undersigned at (858) 505-1166.

Respectfully submitted,

GAZDZINSKI & ASSOCIATES

Dated: August 23, 2001

By: [illegible]

Robert F. Gazdzinski
Registration No. 39,990
3914 Murphy Canyon Road, Suite A232
San Diego, CA 92123
Telephone No.: (858) 505-1166
Facsimile No.: (858) 505-1168

METHOD AND APPARATUS FOR JUMP DELAY SLOT CONTROL IN A PIPELINED PROCESSOR

This application claims priority to U.S. Provisional Patent Application Serial No. 60/134,253 filed May 13, 1999, entitled "Method And Apparatus For Synthesizing And Implementing Integrated Circuit Designs".

Copyright

Related Applications

This application is related to co-pending U.S. Patent Application Serial No. 09/523,871 filed March 13, 2000 and entitled “Method and Apparatus for Jump Control in a Pipelined Processor”, U.S. Patent Application Serial No. 09/524,179 filed March 13, 2000 and entitled “Method and Apparatus for Processor Pipeline Segmentation and Re-Assembly”, U.S. Patent Application Serial No. 09/524,178 filed March 13, 2000 and entitled “Method and Apparatus for Loose Register Encoding Within a Pipelined Processor”, and U.S. Patent Application No. 09/418,663 filed October 14, 1999, entitled "Method and Apparatus for Managing the Configuration and Functionality of a Semiconductor Design”.

Background of the Invention

1. Field of the Invention

The present invention relates to the field of integrated circuit design, specifically to the use of a hardware description language (HDL) for implementing instructions in a pipelined central processing unit (CPU) or user-customizable microprocessor.

2. Description of Related Technology

RISC (or reduced instruction set computer) processors are well known in the computing arts. RISC processors generally have the fundamental characteristic of utilizing a substantially reduced instruction set as compared to non-RISC (commonly known as "CISC") processors. Typically, RISC processor machine instructions are not all micro-coded, but rather may be executed immediately without decoding, thereby affording significant economies in terms of processing speed. This "streamlined" instruction handling capability furthermore allows greater simplicity in the design of the processor (as

APPENDIX A – EXEMPLARY DELAY SLOT VHDL USED FOR SYNTHESIS

```
library ieee.arc;
use ieee.std_logic_1164.all;
use arc.arcutil.all;

entity delay_slot is:

PORT(       ip2iv         ;     in      std_ulogic;
            ip2condtrue   ;     in      std_ulogic;
            ip2i          ;     in      std_ulogic_vector(4 downto 0);
            ip2dd         ;     in      std_ulogic_vector(1 downto 0);
            p2killnext    ;     out     std_ulogic);

end delay_slot;

architecture synthesis of delay_slot is:

        signal      ip2bch        ;     std_ulogic;
        signal      ip2rjmp       ;     std_ulogic;
        signal      ip21pcc       ;     std_ulogic;
        signal      ip2jumping    ;     std_ulogic;
        signal      ip2nojump     ;     std_ulogic;
        signal      ip2killnext   ;     std_ulogic;

begin

        ip2bch <=     '1' WHEN ip2iv = '1'     AND  (ip2i = obcc OR ip2i = oblcc
                                               OR    ip2i = olpcc OR ip2i =
ojcc ) ELSE
                                   '0';

------ Delay slot canceling logic. ------

-- Due to the fact that the LPcc instruction uses its condition code in a
different way to the other branch instruction, two signals are required in the illustrated
embodiment to specify a jump instruction which is jumping, and one for specifying a
jump instruction which is not.

-- regular jump instruction --

        ip2rjmp      <= '1' WHEN ip2iv = '1'   AND  (ip2i    =    obcc
                                               OR   ip2i     =    oblcc
                                               OR   ip2i     =    ojcc ) ELSE
```

```
                                        '0';

-- loop setup instruction --

        ip21pcc        <= '1' WHEN ip2iv = '1'    AND  (ip2i    =      o1pcc) ELSE
'0';

-- jumping/not jumping signals --

        ip2jumping     <= (ip2rjmp AND ip2condtrue) OR (ip21pcc AND NOT
ip2condtrue);
        ip2nojump      <= (ip2rjmp AND NOT ip2condtrue) OR (ip21pcc AND
ip2condtrue);

-- the kill signal itself --
--
-- ip2killnext include p2iv and a full decode for a jump, so can
-- be used in pipect1 without any further decode (apart from en2 of course).


        ip2killnext<=  '1'    WHEN       ((ip2dd = dmk ) AND ip2bch = '1')
                              OR         ((ip2dd = dmnd) AND ip2jumping = '1')
                              OR         ((ip2dd = dmjd) AND ip2nojump = '1')
ELSE
                       '0';

        p2killnext <= ip2killnext;

end synthesis;
```

APPENDIX B – EXEMPLARY SYNTHESIS SCRIPT USED FOR DELAY SLOT SYNTHESIS

```
/* produce result without any optimization */
/* assuming using the LSI Logic 10k library */

analyze - format vhdl - lib ARC       {ARCHOME  + "/arc/vhdl/arcutil.vhdl"}
analyze - format vhdl - lib USER      {USERDIR   + "/vhdl/delay_slot.vhdl"}

elaborate delay_slot -arch "synthesis" - lib USER

compile

write - format db - hierarchy -output db/delay_slot_noopt.db
remove_design -all

/* result with logic optimization */

elaborate delay_slot -arch "synthesis" -lib USER

/* timing for inputs direct from flipflops */
t_in = 1.33

/* create an imaginary 20MHz clock */

create_clock -name ck -period 50

set_input_delay      20      ip2condtrue    -clock ck
set_input_delay      t_in    ip2dd          -clock ck
set_input_delay      t_in    ip21           -clock ck
set_input_delay      t_in    ip2iv          -clock ck

set_output_delay     28      p2killnext     -clock ck

compile
write -format db -hierarchy -output db/delay_slot_opt.db
```

Revised Claims as of August 23, 2001 (Pages 22 to 28b)

1. A method of controlling the execution of instructions within a pipelined processor, comprising:

providing an instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said words comprising a jump instruction;

assigning one of a plurality of values to at least one of said data bits of said at least one jump instruction; and

controlling the execution of at least one subsequent instruction within said pipeline based on said one assigned value of said at least one data bit when said at least one jump instruction is decoded.

2. The method of Claim 1, wherein the act of assigning comprises: identifying a plurality of data bits within said at least one jump instruction; and assigning one of two discrete values to each of said data bits, the combination of said two discrete values representing at least three jump delay slot modes within said processor.

3. The method of Claim 2, wherein the act of controlling the execution based on said discrete values comprises selecting at least one mode from the group comprising:

(i) executing said at least one subsequent instruction under all circumstances;

(ii) executing said at least one subsequent instruction only if a jump occurs; and

(iii) stalling the pipeline or inserting a bubble into the pipeline if a jump occurs.

4. The method of Claim 3, wherein said at least one jump instruction comprises a conditional branch instruction.

5. The method of Claim 1, wherein the act of controlling the execution based on said one assigned value comprises:

(i) executing said at least one subsequent instruction under all circumstances;

(ii) executing said at least one subsequent instruction only if a jump occurs; and

(iii) stalling the pipeline or inserting a bubble into the pipeline if a jump occurs.

6. A processor design synthesized by the method comprising:

inputting information to a first file to include an instruction set having at least one jump instruction, said at least one jump instruction comprising at least one of a plurality of jump delay modes;

defining the location of at least one library file;

generating a script using said first file, said library file, and user input information;

running said script to create a customized description language model; and

synthesizing said design based on said description language model.

7. The method of Claim 6, wherein the act of synthesizing comprises running synthesis scripts based on said description language model.

8. The method of Claim 7, further comprising the act of generating a third file for use with a simulation, and simulating said design using said third file.

9. The method of Claim 8, further comprising the act of evaluating the acceptability of the design based on said simulation.

10. The method of Claim 9, further comprising the acts of revising the design to produce a revised design, and re-synthesizing said revised design.

11. The method of Claim 1, wherein the act of inputting comprises providing a plurality of input parameters associated with said design, said parameters comprising:

(i) a cache configuration; and

(ii) a memory interface configuration.

12. A machine readable data storage device comprising:

a data storage medium adapted to store a plurality of data bits; and a computer program rendered as a plurality of data bits and stored on said data storage medium, said program being adapted to run on the processor of a computer system and synthesize integrated circuit logic for use in a processor having a pipeline and incorporating an instruction set having a at least one branching instruction and a plurality of jump modes associated therewith, said plurality of jump modes comprising at least the following:

(i) executing a subsequent instruction within said pipeline under all circumstances;

(ii) executing a subsequent instruction within said pipeline only if jumping occurs; and

(iii) stalling said pipeline if jumping occurs.

13. The data storage device of Claim 12, wherein said data storage medium is a compact disk-read only memory (CD-ROM), and said plurality of data bits comprises an object representation of said program.

14. A digital processor comprising:

a processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words;

a data interface between said processor core and an information storage device; and

an instruction set comprising a plurality of instruction words, at least one of said instruction words being a jump instruction containing data defining a plurality of jump delay slot modes, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to said at least one jump instruction word within said instruction set.

15. The processor of Claim 14, wherein said plurality of jump delay slot modes comprises at least the following modes:

(i) executing a subsequent instruction within said pipeline under all circumstances;

(ii) executing a subsequent instruction within said pipeline only if jumping occurs; and

(iv) stalling the pipeline if jumping occurs.

16. The processor of Claim 14, wherein said at least one jump instruction comprises a conditional branch instruction having an associated logical condition, the execution of a jump to the address within said information storage device specified by said at least one conditional branch instruction being determined by said logical condition.

17. A digital processor having at least one pipeline and an associated data storage device, wherein the execution of instructions within said at least one pipeline is controlled by the method comprising:

storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a branch instruction directing branching to a first address within said data storage device;

assigning one of a plurality of values to at least one of said data bits of said at least one branch instruction;

decoding said at least one branch instruction including said one value;

determining whether to execute an instruction within said pipeline in a stage preceding that of said at least one branch instruction based on said one value; and

branching to said first address based on said at least one branching instruction.

18. The processor of Claim 17, wherein said data bits comprise binary (base 2) data.

19. The method of Claim 17, wherein said at least one pipeline comprises at least a three stage instruction pipeline comprising instruction fetch, decode, and execute stages.

20. A method of controlling the branching within the program of a multi-stage pipelined digital processor, comprising:

storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a branch instruction directing branching to a first address within said data storage device based on a first parameter;

defining a plurality of jump modes;

assigning at least one of said plurality of jump modes to at least one of said data bits of said at least one branch instruction;

decoding said at least one branch instruction including said at least one data bit; and

determining whether to branch to said first address based on said at least one data bit and said first parameter.

21. The method of Claim 20, wherein the act of defining a plurality of jump modes comprises defining the following modes:

(i) executing a subsequent instruction under all circumstances;

(ii) executing a subsequent instruction only if jumping occurs; and

(iii) stalling the pipeline or inserting a bubble into the pipeline if jumping occurs.

22. An apparatus for synthesizing the design of logic used within a digital processor, comprising:

a central processing unit;

a data storage device operatively coupled to said central processing unit, said data storage device being adapted to store and retrieve a computer program;

an input device adapted to generate signals in response to inputs from a user of said system;

a computer program, stored on said data storage device, said program being adapted to receive said signals and permit said user to:

input information to a first file to include an instruction set having at least one jump instruction, said at least one jump instruction comprising at least one of a plurality of jump delay modes;

define the location of at least one library file;

generate a script using said first file, said library file, and user input information;

run said script to create a customized description language model; and

synthesize said design based on said description language model.

23. A digital processor comprising:

processing means having a multistage data pipeline, said processing means being adapted to decode and execute an instruction set comprising a plurality of instruction words;

means for storing data;

data interface means for transferring data between said processing means and said means for storing data; and

an instruction set comprising a plurality of instruction words, at least one of said instruction words being a jump instruction containing data defining a plurality of jump control means, said plurality of jump control means controlling the execution of instructions within said data pipeline of said processing means in response to said at least one jump instruction word within said instruction set.

24. An apparatus for synthesizing the design of logic used within a digital processor, comprising:

means for processing data;

means for storing data, said means for storing being operatively coupled to said means for processing, and adapted to store and retrieve a computer program;

means for inputting information operatively coupled to said means for processing, said means for inputting being adapted to generate signals in response to inputs from a user of said system;

a computer program, stored on said data storage device, said program being adapted to receive said signals and comprising:

means for inputting information to a first file to include an instruction set having at least one jump instruction, said at least one jump instruction comprising at least one of a plurality of jump delay modes;

means for defining the location of at least one library file;

means for generating a script using said first file, said library file, and user input information;

means for running said script to create a customized description language model; and

means for synthesizing said design based on said description language model.

25. The method of Claim 1, wherein at least one of said plurality of instruction words comprises an op-code and a plurality of fields, each of said fields comprising a plurality of bits, said at least one instruction word being encoded according to the method comprising:

associating a first of said fields with a first data source;

associating a second of said fields with a second data source; and

performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code.

26. The method of Claim 1, further comprising generating a long immediate constant using a single word instruction according to the method comprising:

providing an instruction word having an op-code and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits;

selecting a portion of said plurality of bits of said at least one short immediate value;

shifting all of said bits of said at least one short immediate value using said op-code and only said portion of bits to produce a shifted immediate value; and

storing said shifted immediate value in a register.

27. The method of Claim 25, wherein said plurality of instruction words further comprises at least one instruction word having an op-code and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits, said at least one instruction word with short immediate value being used to generate a long immediate constant according to the method comprising:

selecting a portion of said plurality of bits of said at least one short immediate value;

shifting all of said bits of said at least one short immediate value using said op-code and only said portion of bits to produce a shifted immediate value; and

storing said shifted immediate value in a register.

28. The method of Claim 27, wherein said at least one instruction word having a plurality of fields and said at least one instruction word having a short immediate value comprise the same instruction word(s).

29. The digital processor of Claim 14, wherein said wherein at least one of said plurality of instruction words comprises an op-code and a plurality of fields, each of said fields comprising a plurality of bits, said at least one instruction word being encoded by:

associating a first of said fields with a first data source;

associating a second of said fields with a second data source; and

performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code.

30. The digital processor of Claim 14, wherein said processor is further adapted to generate a long immediate constant using a single word instruction by:

providing an instruction word having an op-code and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits;

selecting a portion of said plurality of bits of said at least one short immediate value;

shifting all of said bits of said at least one short immediate value using said op-code and only said portion of bits to produce a shifted immediate value; and

storing said shifted immediate value in a register.

31. The digital processor of Claim 29, wherein said plurality of instruction words further comprises at least one instruction word having an op-code and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits, said at least one instruction word with short immediate value being used to generate a long immediate constant according to the method comprising:

selecting a portion of said plurality of bits of said at least one short immediate value;

shifting all of said bits of said at least one short immediate value using said op-code and only said portion of bits to produce a shifted immediate value; and

storing said shifted immediate value in a register.

32. The digital processor of Claim 31, wherein said at least one instruction word having a plurality of fields and said at least one instruction word having a short immediate value comprise the same instruction word(s).